RESTARTABILITY REVIEWED

- PURPOSE
- RESTARTABLE APL STATEMENTS
- RESUME OR →RESTART?
- WHAT SHOULD RESTART DO?
    - DEFINITION
    - →⎕LC
    - TIE FILES
- WS CRASH RECOVERY
- RESTARTING N/B-TASKS
    - EXAMPLE: MABRA
    - EXAMPLE: SV TASK SERVER
- B-TASK RESTARTABILITY
- FILE UPDATES AND RESTARTABILITY
    - CONDITIONS
    - ⎕APPEND EXAMPLES
    - ⎕REPLACE EXAMPLES
- ACKNOWLEDGEMENTS

THE SHARP ENVIRONMENT PROVIDES COMPREHENSIVE TASK RECOVERY MECHANISMS SUCH THAT FOR NORMAL LINE-DROPS AND SHUTDOWN DURING AN ACTIVE SESSION, THE WORKSPACE AND FILES ARE KEPT IN SYNCHRONIZATION, AND THE SI STACK POINTS TO THE LINE OF INTERRUPTION.

FOR MOST NON-CRITICAL OPEN APPLICATIONS, IT IS SUFFICIENT TO RESTART BY RETYING FILES AND BRANCHING TO THE CURRENT VALUE OF □LC.

IN THE CASE OF SENSITIVE CODE INVOLVING FILE HOLDS OR SHARED VARIABLES, IT MAY BE NECESSARY TO)RESET AND RESTART FROM A LESS CRITICAL POINT. THUS, FOR MOST SITUATIONS, RESTARTABILITY IS A NON-ISSUE. BUT PROBLEMS MAY STILL CROP UP:

THE FUNCTION LINE TO WHICH THE USER BRANCHES MAY CONTAIN NON-RESTARTABLE CODE.

IN SITUATIONS WHERE LINE DROPS ARE FREQUENT, MANUALLY EXECUTED INITIALIZATIONS MAY LEAD TO ERRORS WHICH COULD HAVE BEEN AVOIDED BY MECHANISMS AUTOMATICALLY EXECUTABLE BY OUR SYSTEM.

NASTY PROBLEMS MAY ALSO ARISE IN ATTEMPTING TO RESTART AFTER SYSTEM CRASHES IN WHICH THE RECOVERED WS MAY LAG THE FILES BY AS MUCH AS FOUR SECONDS. AS THE ACTIVE WSS IN CORE ARE LOST AND HAVE TO BE RETRIEVED FROM SECONDARY MEMORY, RE-ENTRY IN THESE SITUATIONS CAN SOMETIMES LEAD TO DANGEROUS RESULTS, AS THE SI POINTED TO IS FROM A PREVIOUS POSITION IN TIME.

THIS PAPER REVIEWS THE MOST COMMONLY USED RESTARTABILITY TECHNIQUES WITH EXAMPLES TAKEN FROM ACTUAL OPEN(ABLE) SYSTEMS.

THE FUNCTIONS DISPLAYED ARE ONLY MEANT AS EXPOSITORY EXAMPLES TO PARTICULAR PROBLEMS.

EACH SYSTEM'S INDIVIDUAL NEEDS SHOULD DETERMINE THE PHASES OF RESTARTABILITY TO BE CONSIDERED AND THE PARTICULAR METHOD TO BE UTILIZED.

WRITING RESTARTABLE APL STATEMENTS

IF AS A RESULT OF AN INTERRUPTION DURING EXECUTION OF A LINE, A BRANCH TO THE ⎕LC REPORTED BY THE SYSTEM CAUSES THE APPLICATION TO MALFUNCTION, THE STATEMENT IS NON-RESTARTABLE. THIS SITUATION CAN ARISE IN A VARIETY OF WAYS:

PARTICULARLY DANGEROUS IN DROP/CRASH RECOVERY SITUATIONS IS INCREMENTATION OF A VARIABLE FOLLOWED BY
- FILE OPERATIONS
- SHARED VARIABLES
- ⎕DL
- USER DEFINED FUNCTIONS

CONSIDER THE FOLLOWING EXAMPLE:

```
      ∇ NICE
 •
[6]   I←I+1
[7]   MAT ⎕REPLACE FILE,CMP[I]
[8]   ...
 •
      ∇


      ∇ UGH
 •
[7]   MAT ⎕REPLACE FILE,CMP[I←I+1]
[8]   ...
 •
```

ASSUME THAT BOTH FUNCTIONS ARE INTERRUPTED JUST PRIOR TO THE ⎕REPLACE (E.G. IMPATIENT USER WITH SLOW FILE SYSTEM, OR SYSTEM CRASH RECOVERY SITUATION).

→RESTART IN FUNCTION <NICE> WILL WORK AS EXPECTED, WHEREAS IN <UGH> THE MATRIX <MAT> WILL BE PLACED ONE COMPONENT AHEAD OF EXPECTED.

IN WRITING APL CODE IT IS ADVISED TO KEEP VARIABLE RESETTING EXPRESSIONS, IF USED, ON A SEPARATE LINE BY THEMSELVES.

IN CASE SUSPENSION IS NOT DESIRED IN A 'SENSITIVE' AREA, THE LOCAL ⎕LX CAN BE SET TO SIGNAL AN EVENT WHICH WILL CUT IT OUT OF THE CURRENT SUSPENSION TO A MORE SECURE AREA PRIOR TO INTERRUPTING.

```
    ∇ CLEAN;⎕TRAP;⎕ER;⎕LX
[1]   ⎕TRAP←'∘ 707  C     ' ◊ ⎕LX←'SIGNAL 707'
[2]   . . .
[3]   SENSITIVE
[4]   . . .
    ∇

    ∇ SENSITIVE
[1]   . . .
[2]   NON-RESTARTABLE CODE
[3]   . . .
    ∇
```

RESUME OR →RESTART ?

AS MANY USERS ACCESS OUR SYSTEM WITH NON-APL TERMINALS, IN SOME CASES WITH A LIMITED CHARACTER SET (E.G. TTY), IT MAY BE IMPOSSIBLE TO RESTART EVEN IF ONE WISHED TO, FOR LACK OF → OR ⎕.

WITH ⎕TRAP/⎕SIGNAL IT IS NOW POSSIBLE (WITH APOLOGIES TO STRUCTURALISTS) TO SPECIFY A RESUME FUNCTION WHICH WILL EXIT FROM ITSELF AND AUTOMATICALLY BRANCH TO THE SUSPENSION.

SOME VARIANTS:

```
     ∇ RESUME;⎕TRAP
[1]   ⍝DEFAULT 'GLOBAL' ⎕TRAP←'∇ 888 E →RESTART'
[2]    ⎕SIGNAL 888
     ∇
```

```
     ∇ RESUME;⎕LX
[1]   ⎕LX←'→⎕LC' ⍝ DON'T SUSPEND IN RESUME
[2]   X←⎕PACK 'X ⎕TRAP'
[3]   ⎕TRAP←'∇ 888 E ⎕PDEF X ◊ →RESTART'
[4]   ⎕SIGNAL 888
     ∇
```

```
     ∇ RESUME;⎕LX
[1]   ⎕LX←'→⎕LC'
[2]   ⎕SP←⎕PACK '⎕SP ⎕TRAP ⎕ER'
[3]   ⎕TRAP←'∇ 888 E ⎕PDEF ⎕SP ◊ →RESTART'
[4]   ⎕SIGNAL 888
     ∇
```

IN THE LATTER TWO CASES, A CONFUSED STATE WOULD ARISE WITH ⎕SP, WERE THERE TO BE A LINE DROP PRIOR TO THE ⎕SIGNAL. A LOCALIZED ⎕LX ENSURES THAT <RESUME> TERMINATES EXECUTION IN SUCH A SITUATION.

WHAT SHOULD RESTART DO?

```
    ∇ LC←RESTART;⎕TRAP;⎕IO
[1]   ⎕IO←1 ◇ ⎕TRAP←' 0 S ....'
[2]   LC←1↓⎕LC ⍝POINT TO SUSPENSION
[3]   CHECK⍙SI ⍝WARN ABOUT MULTIPLE SUSPENSIONS
[4]   TIE⍙FILES ⍝RETIE PREVIOUSLY TIED FILES
[5]   PROFILE ⍝SET USER PROFILE
[6]   TIE⍙SV ⍝REESTABLISH SHARED VARIABLES
    ∇
```

THE ABOVE STATEMENTS STARTING FROM LINE 2 REFLECT, IN DESCENDING ORDER OF FREQUENCY THE OPTIONS FOUND IN THE <RESTART> MODULES ON OUR SYSTEM.

A NOTE ABOUT LINE 2 - POINTING TO THE SUSPENSION:

THE NEW DEFINITION ⎕LC ON OUR SYSTEM WHICH NOW RETURNS THE ACTUAL LINE NUMBER OF THE FUNCTION IN WHICH WE MAY BE SUSPENDED IN VARIOUS LEVELS OF ⎕, ⍞, AND ⍎, MAKES THE FOLLOWING EXAMPLE OBSOLETE. (ALTHOUGH IT IS STILL FOUND IN MANY APPLICATIONS.)

```
    ∇ LC←RESTART⍙OLD
[1]   LC←(1+ '⍎'=1↑,1 0↓2 ⎕WS 2)↓⎕LC
[2]   ⍝TECHNICAL SUPPLEMENT MAR/APR 76
    ∇
```

IT IS NOW SUFFICIENT TO SPECIFY →⎕LC (OR →1↓⎕LC IF UNDER <RESTART>) TO GET US BACK TO THE START OF THE LINE IN SUSPENSION.

TYING FILES UPON RESTART

AS A SAVED WORKSPACE CONTAINS NO RECORD OF WHAT FILES WERE TIED, OR WHAT NUMBERS WERE USED TO TIE THEM, IT IS NECESSARY TO HAVE A MECHANISM TO KEEP TRACK OF THIS INFORMATION EACH TIME A FILE IS CREATED, TIED, RENAMED, ERASED OR UNTIED.

EXAMPLE

OPEN SYSTEM
LIST OF TIED FILES KEPT IN VARIABLE <FNAMES>
LIST OF TIED FILE NUMBERS IN VARIABLE <FNUMS>

```
    ∇ T←STIE FNM;B;⎕TRAP;⎕ER;⎕IO
[1]   ⍝ FNM IS ANY ACCEPTABLE LEFT ARGUMENT OF ⎕STIE.
[2]     ⎕IO←1 ◇ ⎕TRAP←'∇ 11 19 22 23 26 28 29 30 31 C
        ⎕SIGNAL⍨4⍴⎕ER ∇ 24 C ←TIED'
[3]     FNM ⎕STIE T←((⍳1+⍴FNUMS)≠FNUMS←⎕NUMS)⍳0
[4]     FNAMES←⎕NAMES ◇ FNUMS←⎕NUMS ◇ →0
[5]   TIED:FNM←((~1↑⎕VI FNM)/(¯1↑⎕AI),' '),(~1↑B)↓(B∨1↓
        (B←FNM≠' '),0)/FNM
[6]     T←(⎕NAMES∧.=(¯11+FNM⍳' ')⍴22↑FNM)/⎕NUMS
    ∇

    ∇ A RENAME B
[1]   A ⎕RENAME B ◇ FNAMES←⎕NAMES
    ∇

    ∇ ERASE A
[1]   FNAMES[FNUMS⍳A;] ⎕ERASE A
[2]   FNUMS←⎕NUMS ◇ FNAMES←⎕NAMES
    ∇

    ∇ UNTIE A
[1]   ⎕UNTIE(A∊⎕NUMS)/A ◇ FNAMES←⎕NAMES
[2]   FNUMS←⎕NUMS
    ∇
```

TYING FILES UPON RESTART ———

UPON RESTART THE FUNCTION <TIE∆FILES> CALLED BY THE <RESTART> FUNCTION WILL TIE ALL FILES IF POSSIBLE AS SAVED IN THE VARIABLE LIST <FNAMES> AND <FNUMS>.

```
    ∇ TIE∆FILES;⎕LX;I;⎕TRAP
[1]   ⎕LX←'→1' ⍝   IF CRASH DURING RE-START,
[2]  ⍝   DON'T CONTINUE TO RETIE FILES, START AGAIN
[3]   →(⎕AVAIL '')⍴L1 ◊ I←⎕DL 10 ◊ →⎕LC
[4]  ⍝   WAIT TILL FILES COME UP.
[5]  L1:⎕UNTIE ⎕NUMS
[6]   I←1 ◊ ⎕TRAP←'∇ 19 29 C →ERR'
[7]  ⍝   FILE ACCESS/ VOL CLASS BYPASS
[8]  L2:→(I>⍴FNUMS)⍴0
[9]   FNAMES[I;] ⎕STIE FNUMS[I]
[10] L3:I←I+1
[11]  →L2
[12] ERR:'UNABLE TO TIE ',FNAMES[I;],' DUE TO ',5↓⎕ER[1;]
     ◊ →L3
    ∇
```

WS CRASH RECOVERY

FILE OPERATIONS LIKE ⎕APPEND AND ⎕REPLACE ARE ACTED UPON IMMEDIATELY FROM APL TO SECONDARY STORAGE. ACTIVE WORKSPACES WHICH ARE RESIDENT IN CORE ARE SWAPPED OUT TO SECONDARY STORAGE AT LEAST EVERY 4 SECONDS.

A SYSTEM CRASH LOSES THE 'CURRENT' COPY OF ACTIVE WORKSPACES THAT WERE IN MAIN MEMORY. THE 'WS CRASH RECOVERY' PROCEDURE, RUN AFTER THE SYSTEM IS BROUGHT UP, BUT BEFORE USERS ARE ALLOWED TO SIGN-ON, TRIES TO RETRIEVE THE MOST RECENT COPY OF EACH ACTIVE WORKSPACE FROM SECONDARY MEMORY (SWAP EXTENTS). THIS COPY WAS WRITTEN TO SECONDARY MEMORY WITHIN 4 SECONDS OF THE CRASH AND MAY BE IN THE MIDDLE OF EXECUTING A LINE. AS A WORKSPACE CANNOT BE SAVED INTO 'CONTINUE' IN SUCH A STATE, THE WS IS PUT INTO EXECUTION TO

TRY TO ALLOW FINISHING THE LINE SO THAT IT CAN BE SAVED READY TO EXECUTE THE NEXT EXECUTABLE STATEMENT.

THUS, PRIOR TO SAVING THE WORKSPACE IN 'CONTINUE', 'WS CRASH RECOVERY' EXECUTES THE MOST CURRENT LINE OF THE WS FROM THE POINT OF INTERRUPTION, WITH AN ATTENTION AND BOUNCE SET WITHIN THE FOLLOWING CONSTRAINTS:

1) A MAXIMUM OF 60 CPU UNITS ARE ALLOWED BEFORE AN INTERRUPT IS CAUSED,

2) ANY CALL TO A FILE FUNCTION OR SHARED VARIABLE REFERENCE TRIGGERS AN ERROR, WITH EXECUTION OF THE LINE INTERRUPTED,

3) ⎕ OR ⍞ ARE SET UP IN SI AND EXECUTION IS SUSPENDED.

4) A REFERENCE TO ⎕DL IS CONSIDERED A NO-OP WITH ATTENTION SET.

5) AN APL ERROR OCCURS WHICH TRIGGERS A BOUNCE (NOTE THAT BOUNCE SET DISABLES ANY ACTIVE ⎕TRAPS)

IF THE LINE EXECUTES SUCCESSFULLY, THE)SI IN THE 'CONTINUE' WS WILL POINT TO THE NEXT STATEMENT TO BE EXECUTED. OTHERWISE, THE LINE NUMBER IN THE)SI REMAINS UNCHANGED WITH THE EXCEPTION OF (3), WHERE)SI WILL POINT TO ⎕ OR ⍞, RESPECTIVELY.

THE WORKSPACE IS THEN SAVED INTO 'CONTINUE' WITH THE 'RECOVERED' MESSAGE AND TIME STAMP.

IF ONE OR MORE SWAP EXTENTS ARE UNREADABLE OR OTHERWISE UNUSABLE, THEN WSS ON THOSE EXTENTS ARE NOT RECOVERED, AND THE USER IS LEFT WITH NO CONTINUE WS FROM THIS CRASH.

RESTARTING N/B-TASKS

N/B-TASKS THAT TERMINATE EXECUTION WITHOUT CLEAROUT OR A ⎕TRAP WITH ' 2001 D CLEAR' SET ARE SAVED UNDER THE NAME INDICATED IN THE THIRD FIELD OF THE ⎕RUN PARAMETERS.

ASSUMING A CLEAROUT IS NOT SET, MOST ERRORS CAN BE TRAPPED AND ACTED UPON IMMEDIATELY AT EXECUTION TIME, WITH THE FOLLOWING LIST CONSITUTING NON-TRAPPABLE EVENTS:

1) THE CPU OR CONNECT UNITS SPECIFIED IN THE ⎕RUN PARAMETERS OR USER LIMITS (VIA NUMBREQ) ARE EXCEEDED.

2) A ⎕BOUNCE IS ISSUED

3) A SWAP ERROR OR SYSTEM ERROR OCCURS, WHICH CAUSES THE CONTENTS OF THE WS TO BE CLEARED WITH THE SAVED WS CONTAINING ⎕ER SET TO 67 AND 68 RESPECTIVELY.

IN MOST SITUATIONS LOADING THE SAVE WS ID AND REACTIVATING IT AFTER TAKING THE CORRECTIVE STEPS ARE SUFFICIENT CONDITIONS FOR RESTARTABILITY. NOTE THAT THE ⎕LX SHOULD ALLOW THE S/T-TASK USER THE OPTION OF INTERVENING MANUALLY PRIOR TO EXECUTING THE DEFERRED TASK ⎕LX, WITH ANY ⎕TRAPS DEACTIVATED. IN CASE A DISTINCTION NEEDS TO BE MADE BETWEEN S AND T TASKS

```
S-TASK ← 0=(2 ⎕WS 3)[9+⎕IO]
```

IT IS HOWEVER POSSIBLE TO AUTOMATE SOME OF THE DIAGNOSTIC MECHANISMS TO SIMPLIFY RESTARTABILITY OF N/B-TASKS. ONE SUCH SCHEME USED BY MABRA IS AS FOLLOWS:

RESTARTING N/B TASKS

```
    ∇ UPDATE;PKG;SUBMITREQUEST;T;NUM;HELP;TYP
[1]   ⍝  86 MABRA UPDATE INITIATOR
[2]    . . .
[3]   L2:PUT CR,'* INITIATE UPDATE TASK'
[4]    T←RESTARTUPDATE ⍝ CHECK STATUS OF PREVIOUS UPDATE
[5]    →(CONDITION T)/L4,ERR,ERR1,ERR3 ◊ →0
[6]   ⍝
[7]   ⍝     T←0 UPDATE READY TO RUN
[8]   L4:NUM←(⎕SIZE(UPDTY),P)[2]-(⎕READ(DIRTY,4),P)[2]
[9]    CR,'* ',(⍕NUM),' REQUEST',((1≠NUM)/'S'),' PENDING
       UPDATE.'
[10]   →(0=NUM)⍴0
[11]  L1:HELP←68
[12]   →(0∈⍴TYP← 0 1 ASK CR,'* RUN AS N-TASK OR B-TASK?
       (N/B): ')⍴0
[13]   →(TYP∈'NB')↓ERR2
[14]   . . . . . ◊ SUBMITREQUEST
[15]   →0
[16]  ERR2:CR,'? ENTER ONE OF N OR B' ◊ →L1,IBUF←''
[17]  ⍝
[18]  ⍝ ERROR CODES RETURNED FROM <RESTARTUPDATE>
[19]  ERR:CR,'? UPDATE ',('NB'[T]),'TASK PENDING',IBUF←''◊→0
[20]  ERR1:CR,'? UPDATE TASK IN PROGRESS' ◊ →0,IBUF←''
[21]  ERR3:CR,'? PROBLEMS WITH UPDATE TASK. CONTACT
       CREATOR.',IBUF←'' ◊ →0
    ∇
```

1) ONE FILE COMPONENT CONTAINS THE DEFERRED TASK STATUS.

2) AN UPDATE REQUEST FIRST CHECKS THIS COMPONENT WITH INTERLOCK TO SEE IF THE DEFERRED TASK IS FREE TO RUN.
   - IF IT IS, IT STARTS IT UP
   - IF A TASK IS ALREADY RUNNING, IT NOTIFIES THE USER AND EXITS
   - OTHERWISE, IT ATTEMPTS TO RESTART THE PREVIOUSLY CRASHED TASK AND ISSUES DIAGNOSTIC MESSAGES PRIOR TO GIVING UP.

RESTARTING N/B-TASKS

```
    ∇ R←RESTARTUPDATE;T;RERUN;ST;QRUN
[1]   ⍝ THIS FN REQUIRES EXPLAINING HOW UPDATE TASKS
[2]   ⍝ ARE CHECKED FOR STATUS AND RESTARTED (IF NECESSARY)
[3]    RERUN←0 ⍝RERUN FLAG. SET TO 1 IF UPDATE TASK RESTARTED
[4]   L0:⎕FHOLD 2 1 ⍴(DIRTY),P ⍝HOLD DIR FILE TO CHECK STATUS
[5]   ⍝(UPDATE TASK ALSO HOLDS THIS FILE)
[6]    ST←⎕READ(DIRTY),4,P ⍝ READ UPDATE STATUS COMPONENT
[7]    →(CREATOR≠⎕AI[1])⍴L10
[8]   ⍝ ⍴ST ⟷ 4
[9]   ⍝ ST[1]=0 - UPDATE TASK FREE TO RUN
[10]  ⍝         1 - NTASK PENDING (HOPEFULLY NEVER THE CASE)
[11]  ⍝         2 - BTASK PENDING
[12]  ⍝         3 - ACTIVE NTASK RUNNING UNDER TASK ID ST[3]
[13]  ⍝         4 - ACTIVE BTASK WITH BTASK REQ.NO. ST[3]
[14]  ⍝         5 - FILE RESERVATION ERROR OCCURRED
[15]  ⍝         6 - UPDATE HALTED - POSS MAXNUMRCDS EXCEEDING
[16]   →(ST[1]=0,⍳6)/L10,L1,L11,L2,L5,L7,L8
[17]  L1:ST[1 3]←0 ◊ →L9 ⍝ RESET STATUS INDICATOR
[18]  L11:T←BTINQ ST[3]
[19]   →((T[1;2]∈ 0 3)∧T[1; 3 4]∧.=0)⍴L1
[20]   →(T[1;2]=3)⍴ERR3
[21]   →L10
[22]  L2:→(ST[3]∈⎕RUNS[;1])↓L3 ⍝ NTASK STILL ACTIVE?
[23]   CR,'* UPDATE NTASK STILL ACTIVE' ◊ →L10
[24]  L3:→RERUN⍴ERR2 ⍝ NTASK RESTARTED AND AND CRASHED?
[25]   CR,'? UPDATE NTASK FAILED'
[26]  L4:'  RESTARTING UPDATE AS AN NTASK'
[27]   QRUN←⎕RUN ': ',(14⍴'CUPD',CCURSYSNUM,' '),' 0 0'
[28]  ⍝ RESTART CRASHED WS TO SEE IF IT WILL RESTART
[29]   →(0≠QRUN[1])⍴ERR1 ⍝ ⎕RUN ERROR?
[30]   ST[1 3]←3,QRUN[2] ⍝ CHANGE STATUS IND TO ACTIVE NTASK
[31]   ST ⎕REPLACE(DIRTY),4,P ⍝REPLACE STAT IND FOR UPDATE
[32]   RERUN←1
[33]   ⎕FHOLD⍳0 ⍝ FREE HOLD SO UPDATE TASK CAN USE DIR FILE
[34]   →L0,⎕DL 2 ⍝ WAIT A WHILE TO SEE IF UPDATE IS WORKING
[35]  L5:→(2=(INQ ST[3])[1;2])↓L6 ⍝ BTASK ACTIVE?
[36]   CR,'* UPDATE BTASK STILL ACTIVE' ◊ →L10
[37]  L6:CR,'? UPDATE BTASK FAILED' ◊ →L4
[38]  L7:CR,'? FILE RESERVATION ERROR OCCURRED DURING UPDATE'
[39]   '  PLEASE MESSAGE OPR FOR MORE FILE SPACE'
[40]   ST[1]←3 ◊ →L9 ⍝ FLAG UPDATE TASK AS ACTIVE NTASK
[41]  ⍝SO RESTART WILL BE AUTOMATIC WHEN RE-ENTERING
[42]  L8:CR,'? UPDATE HALTED AS MAX NO OF RECORDS EXCEEDED'
[43]   CR,'? PLEASE RUN THE IN (INVERT/RESIZE) COMMAND'
[44]  L9:ST ⎕REPLACE(DIRTY),4,P
[45]  L10:⎕FHOLD⍳0
[46]   R←1↑ST
[47]   →0
[48]  ERR1:CR,'? ⎕RUN ERROR: ',⍕QRUN ◊ →(12=1↑QRUN)↓L10,L1
[49]  ERR2:CR,'? UNABLE TO RESTART. CONTACT SHARP' ◊ →L10
[50]  ERR3:CR,'?BTASK ERR:',(⍕T[1; 3 4]),'SEE SATN 4 5' ◊ →L1

    ∇
```

RESTARTING N/B-TASKS

EXAMPLE OF SV NTASK MULTISERVER SYSTEM (RE)START MODULES FOR T-TASK USER/N-TASK SERVER.

```
    ∇ R←ΔAWAKEN SVAR;NMS;P;T;TASKNO;I;⎕TRAP
[1]   ⍝ CALLED BY TTASK PROCESSOR.
[2]   ⍝ <SVAR> IS LITERAL VECTOR, THE NAME OF A SHARED
[3]   ⍝ VARIABLE SVAR FOR SHARE WITH THE NTASK.
[4]   ⍝ <R> IS: 1 IF NTASK IS UP;
[5]   ⍝          A LITERAL ERROR MESSAGE OTHERWISE
[6]   ⍝ WILL ATTEMPT TO START NTASK IF NECESSARY
[7]   ⍝ ROPE 7 MAY 1979
[8]    ⎕TRAP←'∘1000 E (5↓⎕ER[1;]) ⎕SIGNAL 599'
[9]    →(0∈⍴NMS←⎕SVQ P← 1234567 99)↓SHR
[10]   →(0=1↑TASKNO←SUBMITNTASK)↓ERR
[11] WT:T←⎕SC ◊ →(1=1↑⍴⎕RUNS)↑ERR ◊ →(2=⎕SVO SVAR)↓WT
[12]   NMS←⎕SVQ P
[13] SHR:→(R←2∧.=(⍴(1↑⍴NMS),2)⍴P) ⎕SVO NMS)↓ERR
[14]   T← 1 1 1 0 ⎕SVC NMS ◊ →OUT
[15] ERR:R←'*UNABLE TO RETRIEVE DATA. PLEASE CALL FOR HELP.'
[16] OUT:
    ∇
```

```
    ∇ CONTROL;ANS;NEXT;SVR;T;⎕LX;CLONE;⎕ER;⎕TRAP
[1]   ⍝ CONTROL FN IN PROCESSOR NTASK. CALLED BY ⎕LX.
[2]   ⍝ ROPE 7 MAY 1979
[3]    ⎕LX←'→(0=⎕RUNS[1;4])↑0 ◊ CR←ΔRESTART ◊ SVR←CRASHED
       ''''' ◊ →CR'
[4]   ⍝ IF ERROR EVENT, ISSUE WARNINGS TO USERS,
[5]   ⍝ REFUSE DATA. OTHERWISE RESTART
[6]   ⍝ GO AWAY IF CLONEID IS TAKEN
[7]    ⎕TRAP←'∘0 E ERR←⎕ER◊⍎NEXT[1;],''←CMSG'' ◊CRASHED SVR'
[8]    →(¯1=CLONE←⎕SVN 99)↑AWAY
[9]    SVR←ΔOFFERS
[10] CHECK:→(⎕SC∧2∈⎕SVO SVR)↓AWAY
[11]   NEXT←ΔPENDING SVR
[12] LOOP:→(0∈⍴NEXT)↑CHECK
[13]   ANS←PROCESS NEXT[1;]
[14]   ⍎NEXT[1;],'←ANS'
[15]   NEXT← 1 0 ↓NEXT ◊ →LOOP
[16] AWAY:CLEAROUT
    ∇
```

RESTARTING N/B-TASKS

MULTI-TASK SV SERVER CRASH RECOVERY/WARNING MONITOR

```
    ∇ SV←CRASHED SV;⎕ER;CLN;NEXT;T;⎕TRAP
[1]   ⍝ NTASK CRASH RECOVERY OR WARNING MONITOR:
[2]   ⍝ CALLED BY ⎕LX OF CRASHED DB RETRIEVAL NTASK.
[3]   ⍝ <SV>, MATRIX OF ALL VARIABLES SHARED. RETURNED IFF
[4]   ⍝ CRASH NOT CAUSED BY ERROR EVENT.
[5]   ⍝ USES GLOBAL CMSG, WARNING CRASH MESSAGE.
[6]   ⍝ ROPE 7 MAY 1979
[7]    ⎕TRAP←'∘0 1000 S'
[8]    →(¯1=CLN←⎕SVN 99)↑AWAY
[9]    →(0∈⍴SV)↓CRI
[10]   SV←ΔOFFERS
[11] CRI:→(1000>1↑⎕FI ERR[1;])↓OUT
[12]  ⍝ ERR IS COPY OF OUTER ⎕ER SET UP BY ⎕TRAP
[13] CHK:→(⎕SC∧2∈⎕SVO SV)↓AWAY
[14]   NEXT←ΔPENDING SV
[15] LOOP:→(0∈⍴NEXT)↑CHK
[16]   ⍎NEXT[1;],'←CMSG'
[17]   NEXT← 1 0 ↓NEXT ◊ →LOOP
[18] AWAY:→ ⍝ NO <CLEAROUT> TO ENSURE ERROR PRESERVED
[19]  ⍝              (MAY BE CALLED DIRECTLY BY DYING NTASK)
[20] OUT:
    ∇
```

RESTARTABILITY OF B-TASKS

B-TASKS HAVE THE OPTION OF RESTARTING AUTOMATICALLY IF A SYSTEM SHUTDOWN OR A SYSTEM CRASH OCCURS WHILE THEY ARE STILL ACTIVE:

> THE PROCESSING SPECIFICATION <NORMRST> WILL RESCHEDULE AN ACTIVE B-TASK TO REACTIVATE AUTOMATICALLY WITH THE WS AND FILES IN SYNCHRONIZATION AFTER A SYSTEM SHUTDOWN.
>
> THE SPECIFICATION <ABNO> RESCHEDULES AN ACTIVE B-TASK WITH NO GUARANTEE AS TO WS AND FILE SYNCHRONIZATION TO RESTART AFTER A SYSTEM CRASH.

NEITHER OPTION GUARANTEES A SPECIFIC TIME AS TO WHEN THEY WILL RESTART, AS THE FOLLOWING NOTES ATTEST.

NO. 1395190 FILED 16.39.50 TUE 5 MAY 1981
FROM AASPY
TO HCA

FOR REASONS THAT ARE QUITE BEYOND ME, THE SCHEDULERS (HGAMDE) ARE RELUCTANT TO MAKE ANY GUARANTEES ABOUT THE SCHEDULE. A WORD WITH AN EX-OPR (AND NOW APL PLAYER OF THE FIRST WATERS) - GEOFF LIDDON - TENDS TO MAKE ME BELIEVE THAT THE BTASK IS NOT GUARANTEED TO BE UP BEFORE THE TTASKS GET GOING. REASON: THE BTASK SCHEDULER IS NOT GENERALLY FIRED UP UNTIL ABOUT 5 MINUTES AFTER THE SYSTEM IS BACK ON ITS FEET. (AND WHO WOULD KNOW THIS BETTER THAN AN OPR!) THE SCHEDULER THEN POKES AROUND TO FIND BTASKS WHICH HAD ABNO SET. THEY MAY THEN GET RESTARTED DEPENDING ON THE CURRENT SYSTEM LOAD AND DEPENDING ON WHETHER (BASED ON CONN LIM) IT IS ESTIMATED THAT THEY ARE LIKELY TO COMPLETE BY 9:00AM (YYZ TIME) - IE.- ARE THEY GONNA BE RUNNING WHEN THE LOAD GETS REALLY HEAVY? IF THE BTASK TO BE RESTARTED IS GONNA BE GRINDING DURING PRIME CPU TIME, THE SCHEDULER MAY RE-SLOT THE TASK TO SOME POINT LATER IN THE DAY. (IE.- DON'T SELL CHEAP CYCLES WHEN YOU CAN SELL EXPENSIVE ONES.)

RESTARTABILITY OF BTASKS

NO. 1396232 FILED 12.47.08 WED 6 MAY 1981
FROM MDE
TO HCA

YES, IT CERTAINLY IS POSSIBLE FOR A TTASK TO SIGN ONTO AN ACCOUNT BEFORE AN INTERRUPTED BTASK IS RESTARTED. FOR ONE THING, THE BTASK SCHEDULER GOES TO SLEEP FOR 15 MIN. WHEN IT'S BROUGHT UP AFTER A CRASH, TO ALLOW THE SYSTEM TO STABILIZE. TASKS WHICH ARE SCHEDULED FOR RESTARTING ARE GIVEN A HIGHER PRIORITY, BUT THAT SIMPLY MEANS THEY OCCUPY A HIGHER POSITION IN THE QUEUE, THE SCHEDULER STILL WILL ONLY SCHEDULE THE TOP N REQUESTS IN THE QUEUE. AS TO WHAT THE MAXIMUM AMOUNT OF TIME A RESCHEDULED TASK CAN LIE DORMANT IS, THE ANSWER IS THEORETICALLY FOREVER(I.E. THERE IS NO MAXIMUM TIME). /MARK

THUS MULTITASK SERVER SYSTEMS ARE BETTER SERVED BY HAVING THE FIRST T-TASK FIRE UP THE DEFERRED TASK SERVER AS AN N-TASK (AS IN THE FOLLOWING EXAMPLE), THAN IN RELYING ON THE AUTOMATIC PRESENCE OF A B-TASK AFTER A SYSTEM CRASH, OR SHUTDOWN.

```
    ∇ ΔSTART;H
[1]   ⍝ DINERS NTASK START-FN
[2]     . . .
[3]     →(0≠STFLAG[1 3 4])/ERR5,ERR6,ERR8
[4]     H←(1 2 ⍴NTASKACC,CLONEID) ⎕SVO 'SV'
[5]     H← 0 0 1 0 ⎕SVC 'SV'
[6]     ⍙(2≠⎕SVO 'SV')/'→STARTNTASK/¯1'
[7]   ⍝     START SERVER IF NOT AROUND
[8]     . . .
[9]   ERR5:'CRITICAL UPDATE PHASE. RETRY',LF ◊ →0
[10] ERR6:'SHARE CAPACITY EXHAUSTED. TRY LATER.',LF ◊ →0
[11] ERR8:ERRMSG ◊ →0
    ∇


    ∇ Z←STARTNTASK;RUN
[1]     Z←0
[2]     RUN←RUNNTASK

[3]     →(0≠1↑RUN)↑ERR1
[4]     'STARTING NTASK ',⍕RUN[2]
[5]     →0
[6]   ERR1:⎕ER←⎕←'? NTASK START FAILED. ⎕RUN ERROR ',⍕RUN
[7]     MAILBUG ◊ Z←1 ◊ →0
    ∇
```

FILE UPDATING AND RESTARTABILITY

BY FILE UPDATES WE ARE RESTRICTING OUR SCOPE TO ⎕APPEND AND ⎕REPLACE OPERATIONS ONLY.

WORKSPACES AND FILES IN APL WORK ASYNCHRONOUSLY, WHICH CAN CAUSE THE TWO TO BE OUT OF STEP AS A RESULT OF A SYSTEM CRASH.

FOR EXAMPLE, A FUNCTION COULD BE WRITING INFORMATION TO A FILE UP TO THE MOMENT OF A CRASH: BUT THE COPY OF THE WORKSPACE RECOVERED FROM SECONDARY STORAGE COULD BE AS MUCH AS 4 SECONDS OUT OF DATE.

THUS, WHEN DESIGNING UPDATING SYSTEMS USING FILES, IT IS IMPERATIVE TO CONSIDER THE POSSIBILITY OF HAVING FILES AHEAD OF THE FUNCTION. IN RESTARTING UPON SIGNING-ON TO THE <RECOVERED> WS, WE WANT TO BE SURE THAT SUFFICIENT CORRECTIVE STEPS ARE BUILT INTO THE UPDATING FUNCTIONS, IF THIS IS POSSIBLE, TO PERMIT AUTOMATIC SYNCHRONIZATION SO THAT EXECUTION RESUMES FROM THE 'RIGHT' POINT.

IF THIS IS NOT POSSIBLE, AS IN THE CASE OF MOST MULTI-USER CONCURRENT ⎕REPLACE SYSTEMS, THE SYSTEM SHOULD DISALLOW RESTART BY A SIGNAL SET IN ⎕LX THAT CUTS TO A SECURE SECTION.

SINGLE USER SYSTEMS, OR SEQUENTIALLY QUEUED MULTI-USER SYSTEMS (E.G. MABRA) ARE GENERALLY RESTARTABLE AFTER CRASH RECOVERY.

IN ⎕APPEND TYPE UPDATES IT IS SUFFICIENT TO RESTART BY CAPTURING THE INDEX OF THE LAST COMPONENT.

IN ⎕REPLACE UPDATES, THE COMPONENTS' ⎕RDCIS CAN BE EXAMINED FOR A CHANGE IN STATUS.

IN MULTI-USER CONCURRENT SYSTEMS, HOWEVER, NO SUCH GENERALIZATIONS CAN BE MADE, ESPECIALLY FOR ⎕REPLACE UPDATES. RESTARTING INVOLVES CHECKING ⎕RDCIS.

THE FOLLOWING PAGES CONTAIN EXAMPLES OF FILE/WS SYNCHRONIZATION, WITH PARTICULAR ATTENTION TO ⎕APPEND PROCESSES WHICH ARE THE EASIEST TO SYNCHRONIZE UNDER PROGRAM CONTROL.

□APPEND UPDATES

A SIMPLE SEQUENTIAL PROCESS IS UPDATED BY MEANS OF □APPEND. THE FILE TO BE MODIFIED IS BUILT UP IN SEQUENTIAL STEPS. THE PROCESS IS 'SIMPLE' AS THE INFORMATION PRODUCED FOR EACH COMPONENT CAN BE DERIVED BY ONLY KNOWING THE COMPONENT NUMBER.

THE APL REPRESENTATION IS AS FOLLOWS:

```
    ∇ SSU;I;LIM;OUTFILE
 .
[3]    I←0+0
[4]   L10:→(LIM<I←I+1)ρL99
[5]    (PROCESS I) □APPEND OUTFILE
[6]    →L10
 .
[9]   L99: ...
```

'I' IS A MONOTONICALLY INCREASING COUNTER RUNNING TO THE UPPER LIMIT OF PROCESSING STEPS 'LIM'.

'PROCESS' REPRESENTS A 'SIMPLE' COMPUTATION WHICH IS A FUNCTION OF I: EACH EXECUTION OF PROCESS I FOR THE SAME I GIVES THE SAME RESULT.

A SYSTEM CRASH COULD CAUSE THE COUNTER IN THE WORKSPACE TO LAG THE FILE COMPONENTS FOR REASONS DISCUSSED EARLIER. WE WOULD LIKE TO DEVISE A MECHANISM SO THAT, UPON BRANCHING TO A LOCAL RESTART POINT (SAY 'GOON') WE RESET THE NECESSARY COUNTERS.

THE 'RESTARTABLE' FUNCTION WOULD THEN LOOK LIKE THIS:

```
    ∇ SSU;I;LIM;OUTFILE;□LX
 .
[3]    □LX←'→GOON,RESTART'
[4]    I←0+0
[5]   L10:→(LIM<I←I+1)ρL99
[6]    (PROCESS I) □APPEND OUTFILE
[7]    →L10
 .
[9]   L99: ...
 .
[11] GOON:I←⍴⍴/2↑□SIZE OUTFILE ◊ →L10
```

AFTER RESTART, →GOON WOULD RESET THE COUNTER 'I' TO THE LAST COMPLETED PROCESS FOUND IN THE FILE. BRANCHING TO L10 WOULD THEN RESUME THE OPERATION FROM THAT POINT. IN THIS EXAMPLE, WE HAVE ASSUMED THE FILE TO BE INITIALLY EMPTY.

□APPEND UPDATES

GENERALIZING THE SIMPLE SEQUENTIAL PROCESS TO A FIXED (K) NUMBER OF □APPENDS FOR EACH PROCESSING STEP I, THE CORRESPONDING APL FUNCTION, WITH THE RESTART LINE WOULD LOOK AS FOLLOWS:

```
    ∇ SSU;I;K;LIM;N;CMP;□IO
[1]   □IO←1
 .
[3]   I←0+0 ◊ □LX←'→GOON,RESTART'
[4]  L10: J←0+0 ◊  →(LIM<I←I+1)ρL99
[5]  L20: →(N<J←J+1)ρL10  ⍝<N> SUBSTEPS
[6]   (I PROCESS J) □APPEND OUTFILE
[7]   →L20
[8]  L99: ...
 .
[10] GOON: CMP←⍴⍴/2↑OUTFILE  ⍝ NO. OF CMPS
[11]  I←⌈CMP+N  ⍝<I> IS LAST PROCESS IN FILE
[12]  J←1+N|CMP-1  ⍝<J> IS CURRENT SUBSTEP IN FILE
[13]  →L20
    ∇
```

THE 'PROCESS' FOR EACH <I> CAN NOW BE VISUALIZED AS CONSISTING OF N SUB-PROCESSES <J> RESULTING IN A TOTAL OF N×LIM □APPEND'S.

□APPEND UPDATES

COMPLEX SEQUENTIAL SYNCHRONIZATION

A COMPLEX □APPEND OPERATION IS ONE WHERE EACH PROCESS STEP PRODUCES AN ARBITRARY NUMBER OF □APPENDS.

A TYPICAL EXAMPLE OF THIS WOULD BE A REPORT GENERATING PROGRAM WITH □OUT.

EXAMPLE 1.

FIX A □OUT FILE UPON RECOVERY SO THAT THE FILE CONTAINS ALL THE □APPENDS (IF ANY) REQUIRED BY THE □OUT.

```
    ∇ FIXQOUTFILE;COMP;FIRST;FSIZE;I;LAST;T;TY;WAS;QOUT
[1]   ⍝ DROP LAST FEW CMPS IF SYSTEM CRASHED DURING EXECUTION
[2]   ⍝ OF MULTIPLE □APPENDS WHEN □OUT IS SET → FILE
[3]   ⍝
[4]   ⍝ QOUT RESET TO MAINTAIN RESTARTABILITY OF THIS FN
[5]   ⍝ NOTE: QOUT IS A LOCAL VARIABLE. AND □OUT WAS SET TO
[6]   ⍝ ITS ORIGINAL STATE PRIOR TO THE CALL OF THIS FN
[7]    WAS←□OUT QOUT← 0 0
[8]   ⍝ EXIT IF □OUT DOES NOT WRITE → FILE OR FILE NOT TIED
[9]    →((~TY∈□NUMS)∨0=TY←WAS[2])ρL3
```

3027275 DATE 05/07/81,

```
[10] ⍝ SEARCH THE FILE FROM THE TOP TO FIND A HSP MSG
[11]  FIRST←1↑FSIZE←□SIZE TY ◊ I←LAST←¯1+FSIZE[2]
[12] ⍝ →L3 IF NO CONTROL MESSAGE FOUND
[13] L1:→(FIRST>I)ρL3
[14]  →(□AV[256]=1↑,COMP←□READ TY,I)ρL2
[15] ⍝ FREE STORAGE. LOOP AT NEXT COMPONENT
[16]  COMP←'' ◊ I←I−1 ◊ →L1
[17] ⍝ ENOUGH PARAMETERS FOR CONTROL MESSAGE?
[18] L2:→(LAST≥I+□FI 6↓10↑COMP)ρL3
[19] ⍝ DROP COMPONENTS
[20]  □DROP TY,I−LAST+1
[21] L3:T←□OUT QOUT←WAS
    ∇
```

□APPEND UPDATES

EXAMPLE 2. A PRINT MODULE OF N STEPS PRODUCING VARYING AMOUNTS OF OUTPUT AT EACH STEP.

THE FIRST COMPONENT OF THE FILE IS RESERVED AS A STATUS BIT CONTAINING THE LATEST PROCESS STEP I AND THE NEXT FREE COMPONENT NUMBER.

```
    ∇ OUTPUT RPT;□LX;□TRAP;□ER;□IO;I;LIM;NO
[1]   □IO←1 ◊ □TRAP←'* 901 C →GOON,RESTART' ◊ □LX←'RORC'
[2]   I←0+0
[3]   FCLEAR QOUT
[4]  L5:LIM←ρRPT←,RPT
[5]  L10:→(LIM<I←I+1)ρL99
[6]   PRINT I A REPORT MODULE WITH MULTIPLE □APPENDS
[7]   PUTΔSTATUS I A TAGS CURRENT FILE STATE →FILE
[8]   →L10
[9]  L99:PUTΔSTATUS 0 0 ρ'' APROCEED WITH HSP ETC.
[10] GOON:→(0=ρNO←,GETΔSTATUS)ρL99
[11] ANO IS LAST STATUS ON FILE, 1↑1↓□SIZE AT THAT POINT
[12]  □DROP QOUT,NO[2]-1↑1↓□SIZE QOUT
[13]  I←NO[1] ◊ →L5 AIN CASE OF CRASH BEFORE L10
    ∇

    ∇ Z←GETΔSTATUS;□TRAP;□ER
[1]   □TRAP←'0 20 E 0 0 FAPPEND QOUT' ◊ →L1
[2]  L1:Z←□READ QOUT,1
    ∇

    ∇ PUTΔSTATUS I
[1]   →(0≠ρ,I)ρL1 ◊ '' FREPLACE QOUT,1 ◊ →0
[2]  L1:(I,1↑1↓□SIZE QOUT) FREPLACE QOUT,1
    ∇

    ∇ RORC
[1]  A RESTARTING AFTER DROP/CRASH

[2]   'EXAMINE WS CONTENTS + SAVED MESSAGE'
[3]   'TO RESTART UNDER PROGRAM CONTROL'
[4]   'TYPE RESUME IF NORMAL DROP'
[5]   'TYPE RECOVER IF CRASHED WS'
    ∇

    ∇ RECOVER
[1]   SIGNAL 901
    ∇
```

```
      ⎕APPEND UPDATES

    ∇ TABELL NOS;I;LIM;CMP;⎕IO;J;⎕ER;⎕TRAP;⎕LX
[1]   ⍝ THIS FUNCTION SPOOLS REPORTS <NOS>
[2]   ⍝ TO AN HSP FILE <QOUT>, SEPARATED BY PAGINATION
[3]   ⍝
[4]    ⎕IO←1 ◇ ⎕TRAP←'* 901 C →GOON,RESTART' ◇ ⎕LX←'RORC'
[5]   LIM←⍴NOS←,NOS ◇  FCLEAR QOUT ⍝CLEAR ⎕OUT FILE
[6]   L10:→(LIM<I←I+1)⍴L99
[7]    PAGE QOUT ⍝HSP PAGINATION SUBSTEP J←1
[8]   L20:⎕←REPORT I ⍝REPORT SUBSTEP J←2
[9]    →L10
[10]  L99:SUBMIT ◇ →0
[11]  GOON:CMP←⍴⍴←/2↑⎕SIZE QOUT
[12]   I←⌈CMP÷2 ⍝I: LST PROCESS IN FILE
[13]   J←1+2|CMP-1
[14]  ⍝J←1 IS A PAGE DONE, J←2 REPORT OK
[15]   LIM←⍴NOS←,NOS ◇ →(L20,L10)[J]
      ∇

    ∇ SPOOL NOS;I;J;LIM;CMP;⎕IO;⎕TRAP;⎕LX;FST;LST;QAI;⎕ER;K
[1]   ⍝MULTI-CRT-USERS SPOOL REPORTS (~40-200)
[2]   ⍝CHRONOLOGICALLY TO FILE
[3]   ⍝A SEPARATE TASK PRINTS REPORTS ON REMOTE TERMINAL
[4]   ⍝
[5]    ⎕IO←1 ◇ ⎕TRAP←'* 901 C →GOON,RESTART' ◇ ⎕LX←'RORC'
[6]   L5:LIM←⍴NOS←,NOS ◇ I←0÷0
[7]   L10:→(LIM<I←I+1)⍴L99
[8]    PAGE QOUT ⍝ PAGINATION (J←1)
[9]   L20:⎕←REPORT NOS[I] ⍝REPORT (J←2)
[10]   →L10
[11]  L99:→0
[12]  GOON:CMP←2↑⎕SIZE QOUT
[13]   FST←1↑CMP ◇ LST←K←¯1+CMP[2] ◇ QAI←1↑⎕AI
[14]  L80:→(FST>K)⍴L5 ⍝ NO RECORDS OF THIS USER FOUND
[15]   →(QAI=1↑1↓⎕RDCI QOUT,K)⍴L90
[16]   K←K-1 ◇ →L80
[17]  L90:J←1+⎕AV[256]=1↑,⎕READ QOUT,K
[18]   LIM←⍴NOS←,NOS ◇ →(L10,L20)[J] ⍝ J IS SUBSTEP IN FILE
      ∇
```

□REPLACE UPDATES

SIMPLE RANDOM UPDATES CONSIST OF □REPLACE PROCESSES, WHICH CAN BE CHARACTERIZED AS FOLLOWS:

```
    ∇ SRU;I;LIM;OUTFILE
 .
[3]    I←1
[4]   L1:(PROCESS I) □REPLACE OUTFILE, COMP I
[5]    →(LIM≥I←I+1)↑L1
 .
    ∇
```

'I' IS A MONOTONICALLY INCREASING COUNTER RUNNING TO THE UPPER LIMIT OF PROCESSING STEPS 'LIM'.

UNLIKE □APPEND, WE HAVE TO SPECIFY AN OUTPUT LOCATION FOR EACH STEP. THIS IS DONE THROUGH THE FUNCTION 'COMP'.

FOR THE PROCESS TO BE 'SIMPLE' WE FURTHER REQUIRE 'COMP' AND 'PROCESS' TO BE FUNCTIONS OF I. EACH EXECUTION OF 'COMP I' AND 'PROCESS I' FOR THE SAME I MUST GIVE THE SAME RESULTS, RESPECTIVELY. E.G.

```
    ∇ NUMRPLC←REPLACE FLDS;I;UCNS;ΔNMS;PKG
[1]   ⍝ 85 MABRAUTIL - NON-INVERTED FILE UPDATE SEGMENT
[2]   ⍝ ...
[3]    I←1
[4]   LOOP1:→(I>⍴UCNS)⍴L2
[5]    PKG←□READ MASTY,UCNS[I],P ⍝ READ IN MASTER BLOCK
[6]    ΔNMS □PDEF PKG ⍝DEFINE THE FIELDS TO BE CHANGED .
[7]   ⍝ SET THE VARIABLES NEEDED FOR FUNCTION 'XEQ'
[8]    MASK←UCNS[I]=CNS
[9]    SIND←MASK/⍳⍴MASK ⍝       PROCESS I
[10]   TIND←MASK/ROWS
[11]   XEQ
[12]  ⍝ PUT THE MASTER BLOCK BACK
[13]   (PKG □PINS □PACK ΔNMS) □REPLACE(MASTY,UCNS[I]),P
[14]   T←□EX ΔNMS ⍝ EXPUNGE THOSE FIELDS DEFINED
[15]   →LOOP1,I←I+1 ⍝LOOP TO UPDATE NEXT MASTER FILE CMP
[16]  ⍝ ...
    ∇
```

UPON RESTART, WE CANNOT SIMPLY LOOK AT THE END OF THE OUTPUT FILE TO RESET THE COUNTERS. INSTEAD, IN ORDER TO AVOID REPEATING □REPLACE'S FROM THE POINT OF INTERRUPTION, WE CAN START AT THE CURRENT WORKSPACE LOCATION AND MOVE STEP-BY-STEP UNTIL WE GET A NON-INCREASING □RDCI IN THE OUTPUT FILE. THIS WILL INDICATE AN INTERRUPTION IN OUR SEQUENCE OF UPDATES.

REPRESENTING THE PROGRAM IN APL, WE GET THE FOLLOWING:

```
    ∇ SRU;I;LIM;OUTFILE;T1;T2;⎕IO
[1]   ⎕IO←1
 •
[3]   I←1
[4]  L1:(PROCESS I) ⎕REPLACE OUTFILE, COMP I
[5]  L2:→(LIM≥I←I+1)↑L1
[6]  L3:→0
 •
[8]  GOON: →(I∈1,LIM)↑L1
[9]  ⍝ I=1: NO TIMESTAMP FOR TESTING IS GUARANTEED
[10] ⍝ I=LIM: THERE IS NO SUCCESSOR TO THE LAST ITERATION
[11]  T1←(⎕RDCI OUTFILE, COMP I)[3]
[12] L99:→(LIM<I←I+1)↑L3
[13]  T2←(⎕RDCI OUTFILE, COMP I)[3]
[14]  →(T2<T1)↑L1 ◊ T1←T2 ◊ →L99
    ∇
```

ALL OF THE ABOVE EXAMPLES ASSUME THAT

(1) NO FILE COMPONENTS WERE DAMAGED,

(2) THE OUTPUT FILE IS UPDATED THROUGH A SEQUENTIAL INTERLOCK.

□REPLACE UPDATES

EXAMPLE: POINTER-DIRECTORY □REPLACE SYSTEM RESTARTABILITY

CONSIDER A CASE FILING SYSTEM (STARS, MAIL, FUNCTIONS ON A FILE) CONSISTING OF:

A DIRECTORY <DIR>, OF SAY 2×N WHERE
ROW 1 = CASE NUMBERS
ROW 2 = FILE LOCATIONS
WITH ALL ADDRESSABLE COMPONENTS SPECIFIED AND 'FREE POSITIONS' IDENTIFIED BY A ¯1 IN THE CASE NUMBER.

A PROPER SEQUENCING OF OPERATIONS CAN ENSURE THAT A USER ENTERING OR DELETING A CASE WOULD NEVER GET THE WRONG INFORMATION AS A RESULT OF A CRASH.

THIS IS ACHIEVED BY WRITING DATA BEFORE DIRECTORIES, AND DELETING DIRECTORIES BEFORE DATA. BY THIS PROCESS THE USER WOULD BE AT MOST FACED WITH THE INCONVENIENCE OF RE-ENTERING THE LAST CASE IF CAUGHT BY A CRASH WHILE ENTERING INFORMATION.

```
      ⎕REPLACE UPDATES
```

```
    ∇ ADD∆CASE;DIR;LOC;PKG;ROW
[1]   PKG←COLLECT∆CASE
[2]   ⎕HOLD FILE
[3]   DIR←⎕READ FILE,LDIR
[4]   LOC←DIR[2;ROW←DIR[1;]⍳¯1]  ⍝<LOC> IS FILE LOCATION
[5]   DIR[1;ROW]←CASEID PKG ⍝<CASEID> ASSIGNS A CASE NUMBER
[6]   ⍝
[7]   PKG ⎕REPLACE FILE,LOC
[8]   DIR ⎕REPLACE FILE,LDIR
[9]   ⍝
[10]  'CASE ';DIR[1;ROW];' FILED ';STAMP ⎕RDCI FILE,LOC
[11]  ⎕HOLD '' ◊ →0
    ∇
```

- IF THE SYSTEM CRASHES BEFORE LINE 7, THE USER WILL HAVE TO REENTER THE CASE

- IF TEE SYSTEM CRASHES AFTER COMPLETING LINE 7, A RESTART MODULE CAN CAPTURE THE FILED INFORMATION.

- IF THE SYSTEM CRASHES AFTER LINE 8, THE CASE IS FILED AND AVAILABLE.

THE SYNCHRONIZATION FUNCTION REQUIRED WOULD APPEAR AS FOLLOWS (ASSUMING IT WERE KEPT SEPARATE FROM THE ADD∆CASE FUNCTION):

```
    ∇ FIX∆ADD∆CASE;DIR;LOC;ROW;TDIR;TCAS;PKG;⎕LX;⎕IO
[1]   ⎕IO←1 ◊ ⎕LX←'→1,RESTART'
[2]   ⎕HOLD FILE
[3]   DIR←⎕READ FILE,LDIR
[4]   TDIR←¯1↑⎕RDCI FILE,LDIR ⍝ DIRECTORY TS
[5]   LOC←DIR[2;ROW←DIR[1;]⍳¯1] ⍝ LOC IS NEXT <FREE> LOC
[6]   TCAS←¯1↑⎕RDCI FILE,LOC ⍝ <FREE> LOC TS
[7]   →(TCAS<TDIR)⍴L99 ⍝ CASE NOT IN FILE
[8]   DIR[1;ROW]←CASEID ⎕READ FILE,LOC ⍝ UPDATE DIRECTORY
[9]   DIR ⎕REPLACE FILE,LDIR
[10]  'CASE ';DIR[1;ROW];' FILED ';STAMP TCAS
[11] L99:⎕HOLD ''
    ∇
```

ACKNOWLEDGEMENTS

THANKS TO THE FOLLOWING PEOPLE FOR THEIR CODE, CRITICISMS AND FEEDBACK:

DAW, DAK, EBI, EDS, EMC, GOTT, HHY, JHC, JHS, KMC, KWI, MDE, MIKE, MVI, NMS, PESCH, SGA, SJT